```java
// Struttura di un programma Java.
// Un programma in Java è un insieme di dichiarazioni di classi.
// Una classe non può contenere direttamente delle istruzioni, ma può contenere
// la dichiarazione di metodi, che contengono dichiarazioni ed istruzioni
// (terminate da ;).


public class Main {
    // La classe Main contiene un metodo, che si chiama main. In Java ogni metodo
    // può essere   chiamato (invocato) solo all'interno di un altro metodo. Quindi
    // occorre un metodo iniziale per far partire l'esecuzione del programma.
    // Tale metodo iniziale è il metodo main, il quale
    // deve essere sufficientemente generale. L'intestazione del metodo main è
    // public static void main (String[] args)
    // ed è sempre la stessa1.

    // Il metodo main si caratterizza per le seguenti proprietà:
    // - è pubblico, ovvero visibile da ogni altro punto del programma;
    // - è statico, in quanto all'inizio non sono stati creati ancora degli
    // oggetti;
    // - non restituisce niente, in quanto non vi è alcun metodo che lo chiama a
    // cui restituire un valore;
    // - i dati in ingresso sono visti come array di stringhe (oggetti della
    // classe predefinita String).

    // Sappiamo che da una classe possiamo ottenere molteplici istanze e per
    // ciascuna istanza si hanno variabili dai nomi identici ma dai valori
    // distinti (forse "che puntano a locazioni di memoria diverse"
    // sarebbe una definizione più chiara). Se poi vogliamo che una variabile sia
    // la medesima per tutte le     istanze di una classe sappiamo che la dobbiamo
    // invece definire come static.
    // La keyword static in java viene usata per definire una proprietà di
    // oggetti e metodi che sono condivisi da più istanze di una stessa classe.
    // Ciò significa che questo tipo di proprietà non è riferita ad una
    // istanza della classe, ma bensi alla classe stessa. Infatti una modifica ad
    // una variabile statica riflette su tutte le istanze di quella classe.
    // Per i metodi avviene sostanzialmente la medesima cosa:
    // - i metodi non statici sono associati ad ogni singola istanza di una
    // classe e perciò il loro contesto di esecuzione (quindi l'insieme delle
    // variabili cui possono accedere) è relativo all'istanza stessa.
    // - in contapposizione i metodi statici non sono associati ad una istanza
    // ma solo ad una classe. Quindi non potranno interagire con le variabili di
    // istanza, ma solamente con quelle statiche.

    // void è una keyword ("parola chiave") ed è quindi una parola riservata del
    // linguaggio che non può essere usata per altri scopi, In Java la parola
    // chiave void in effetti ha principalmente un solo ed unico uso: dichiarare
    // che un metodo non restituisce alcun valore.

    // Il parametro args rappresenta l'Array contenente tutti i parametri
    // passati al programma (ad esempio dalla linea di comando).

    public static void main(String[] args) {
        // L'operatore new serve a creare una nuova istanza di un oggetto
        // appartenente ad una determinata classe e, conseguentemente,
        // viene allocata automaticamente la memoria necessaria per conservare
        // tale istanza. L'operatore new, per eseguire la creazione di un oggetto,
        // invoca il costruttore della classe che si desidera istanziare.

        Array Elenco = new Array();

        //Dichiaro le variabili necessarie
        final int NMAX = 10;          //non può essere più assegnato
        int vett[]  = new int[NMAX];
        int Trovato;

        System.out.println("attraversa Array VUOTO ----------------------");
        Elenco.attraversaArray(vett);
        System.out.println();

        System.out.println("popola Array e visualizza --------------------");
        Elenco.popolaArray(vett,5,10);
        Elenco.attraversaArray(vett);
```

```
74          System.out.println();
75
76          System.out.println("inserisce Item 22 in 3 in Array e visualizza----------");
77          Elenco.inserisceItemArray(vett,3,22);
78          Elenco.attraversaArray(vett);
79          System.out.println();
80
81          System.out.println("modifica Item in 3 con 23 in Array e visualizza
            ----------");
82          Elenco.modificaItemArray(vett,3,23);
83          Elenco.attraversaArray(vett);
84          System.out.println();
85
86          System.out.println("ricerca LINEARE di Item 23 in Array e visualizza ");
87          Elenco.ricercaItemArrayL(vett,23);
88          Elenco.attraversaArray(vett);
89          System.out.println();
90
91          System.out.println("elimina Item in 3 in Array e visualizza ----------");
92          Elenco.eliminaItemArray(vett,3);
93          Elenco.attraversaArray(vett);
94          System.out.println();
95
96          System.out.println("svuota Array --------------------------------");
97          Elenco.svuotaArray(vett);
98          Elenco.attraversaArray(vett);
99          System.out.println();
100     }
101 }
102
103
104 class Array {
105     // METODO popola Array
106     public void popolaArray (int vett[] , int NumItem, int base) {
107         for(int i=0; i<NumItem; i++)
108             vett[i] = i*base;
109     }
110
111     // METODO attraversa Array
112     public void attraversaArray (int vett[]) {
113         for(int i=0; i<vett.length; i++)
114             System.out.println(i + " " + vett[i]);
115     }
116
117     // METODO inserisce Array
118     public void inserisceItemArray (int vett[],int k,int item) {
119        int i = vett.length-2;
120         while(i >= k) {
121             vett[i+1] = vett[i];
122             i--;
123         }
124         vett[k] = item;
125     }
126
127     // METODO modifica Item Array
128     public void modificaItemArray (int vett[],int k,int item) {
129         vett[k] = item;
130     }
131
132     // METODO elimina Item Array
133     public void eliminaItemArray (int vett[],int k) {
134         int i;
135         for(i=k; i < vett.length-1; i++)
136             vett[i] = vett[i+1];
137     }
138
139     // METODO svuota Array
140     public void svuotaArray (int vett[]) {
141         for(int i=0; i<vett.length; i++)
142             vett[i] = 0;
143     }
144
145     //RICERCA LINEARE
```

```
    public void ricercaItemArrayL (int vett[],int item) {
        int loc;
        int Trovato = -1;
        for(loc=0; loc<vett.length; loc++) {
            if(vett[loc] == item) {
                System.out.println("Elemento trovato in posizione: " + loc);
                Trovato=1;
                break;
            }
        }
        if (Trovato == -1) {
            System.out.println("Elemento non trovato.");
        }
    }

    //RICERCA LINEARE con restituzione
    int ricercaItemArrayLr (int vett[],int item) {
        int loc;
        int Trovato = -1;
        for(loc=0; loc<vett.length; loc++) {
            if(vett[loc] == item) {
                Trovato=loc;
                break;
            }
        }
        return Trovato;
    }
}


OUTPUT
attraversa Array VUOTO ----------------------
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0

popola Array e visualizza --------------------
0 0
1 10
2 20
3 30
4 40
5 0
6 0
7 0
8 0
9 0

inserisce Item 22 in 3 in Array e visualizza----------
0 0
1 10
2 20
3 22
4 30
5 40
6 0
7 0
8 0
9 0

modifica Item in 3 con 23 in Array e visualizza ----------
0 0
1 10
2 20
3 23
4 30
```

```
5 40
6 0
7 0
8 0
9 0

ricerca LINEARE di Item 23 in Array e visualizza
Elemento trovato in posizione: 3
0 0
1 10
2 20
3 23
4 30
5 40
6 0
7 0
8 0
9 0

elimina Item in 3 in Array e visualizza ----------
0 0
1 10
2 20
3 30
4 40
5 0
6 0
7 0
8 0
9 0

svuota Array ---------------------------------
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
```